Class Activity 02 : **Video Coding**

Prepared by **Dr. Imran**

Discrete Cosine Transform of an Image

Goals:

How to find discrete cosine transform and inverse discrete cosine transform of an image using OpenCV Python

Steps

To find discrete cosine transform of an input image, you could follow the steps given below

1. Read the input image using `cv2.imread()` method. The image is in the same directory as Notebook.
2. Convert the input image to *gray scale image* using `cv2.cvtColor()` method.
3. Convert the gray scale image to *np.float32*.
4. Find the *discrete cosine transform* of the image using `cv2.dct()`. {*This method takes a gray scale image in floating point.*}

- Pass flag `cv2.DCT_INVERSE` or `cv2.DCT_ROWS` to the `cv2.dct()` function.
- Visualize the discrete transform of the input image using `plt.imshow()` method.

5. To visualize the input image back after the discrete cosine transform

a. Apply inverse discrete cosine transform `cv2.idct()`
b. Convert the image to *np.uint8*.

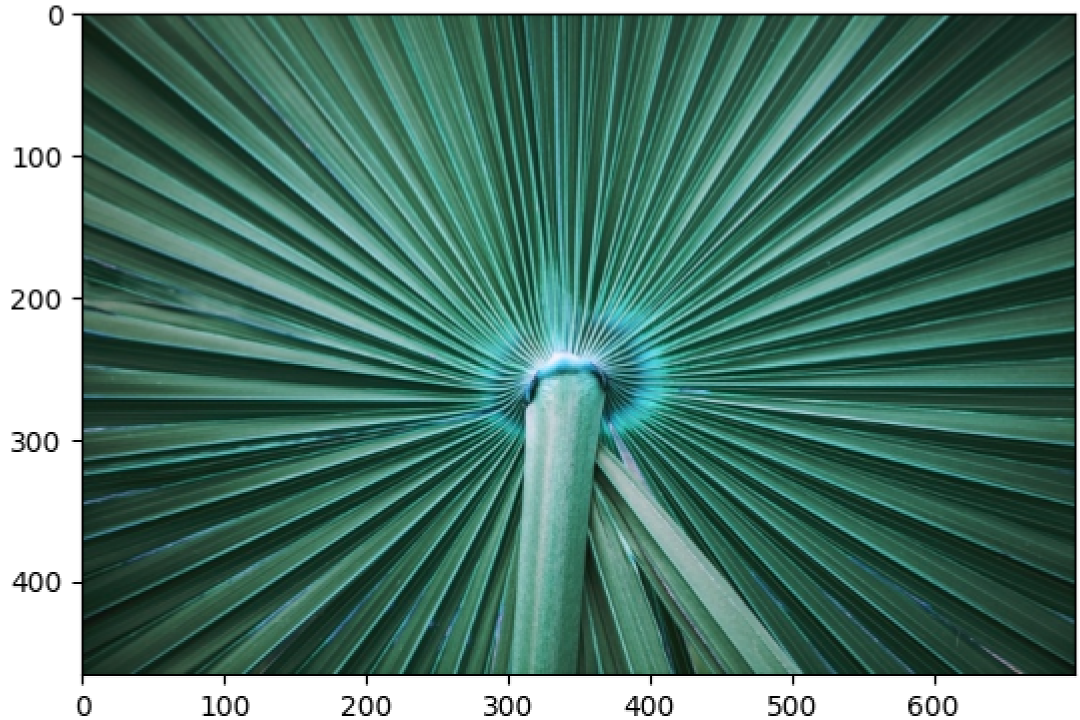
In [1]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

STEP I: Input Image

Use the given image `imageIn.jpg` as the input

1. Read the image
2. Plot the image using `plt.imshow()`
3. Display image shape and data type

In [2]:
```
# Start Script below
#
# YOUR CODE HERE
image = cv2.imread('imageIn.jpg')
plt.imshow(image)
plt.show()

#
# End Script
```

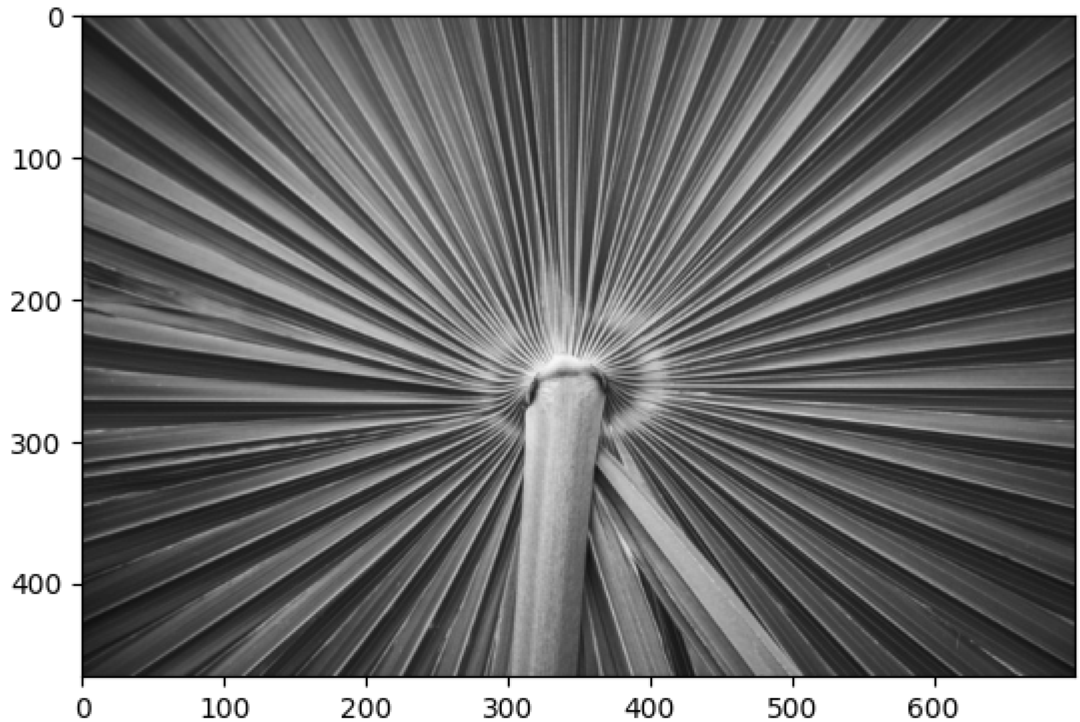


STEP II: Conversion to Gray-Scale

In this step

1. Convert this image into gray scale using `cv2.cvtColor()`
2. Plot the gray scaled image using `plt.imshow()` using `cmap='gray'`, `vmin=0`, `vmax=255` as additional options
3. Display image shape and data type

In [3]:
```
# Start Script below
#
# YOUR CODE HERE
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
plt.imshow(image_gray, cmap='gray',vmin=0, vmax=255)
plt.show()
#
# End Script
```



STEP III: Apply DCT and inverse DCT

In this step,

- Convert the gray scale image into `np.float32()`.
- Find *discrete cosine transform* of the gray-scaled image by using `dct = cv2.dct(gray_image, flag)` two times as:

1. Pass a *flag* `cv2.DCT_INVERSE` to the DCT function `dct1 = cv2.dct()`
2. Pass a *flag* `cv2.DCT_ROWS` to the DCT function `dct2 = cv2.dct()`

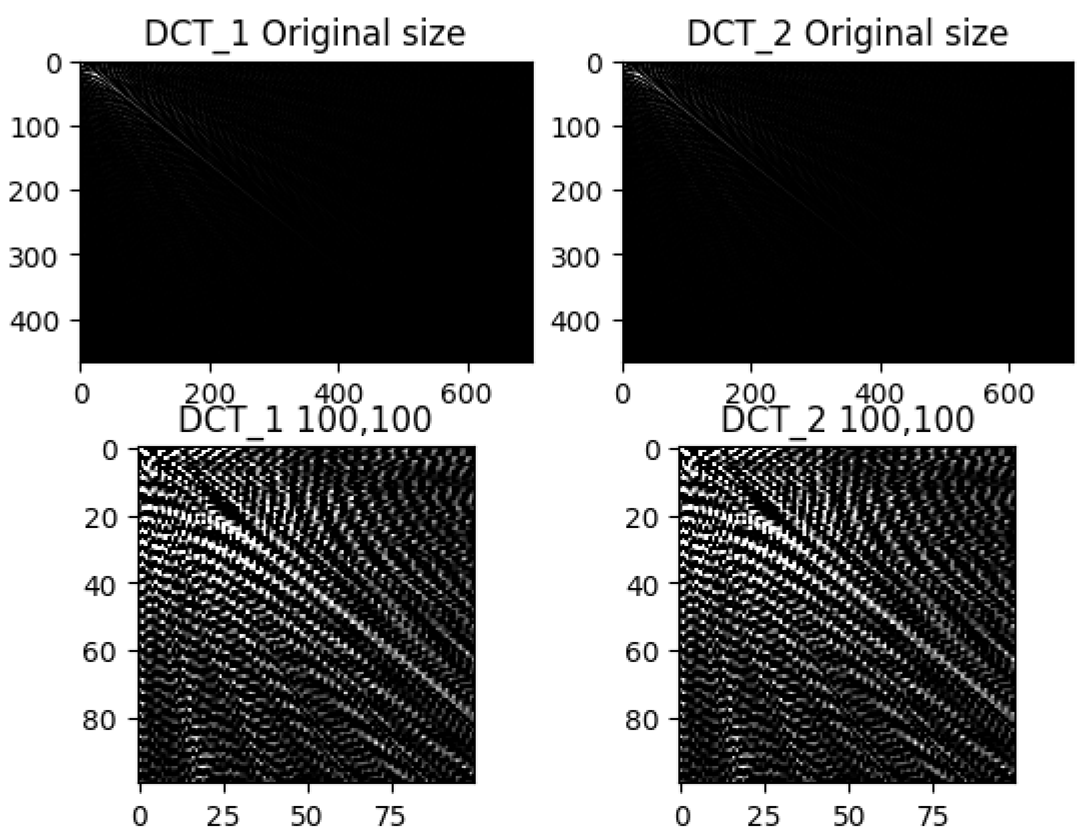
- Plot the resultant images in *subplot* using

`plt.subplot(2,2,1)` for **dct1** (Original Size)
`plt.subplot(2,2,2)` for **dct2** (Original Size)
`plt.subplot(2,2,3)` for **dct1** (100 by 100)
`plt.subplot(2,2,4)` for **dct2** (100 by 100)
3. Display image shape and data type of `dct1` (or `dct2`) image

In [4]:
```
# Start Script below
#
# YOUR CODE HERE
image_gray_reconstruted = np.float32(image_gray)

dct1 = cv2.dct(image_gray_reconstruted, cv2.DCT_INVERSE)
dct2 = cv2.dct(image_gray_reconstruted, cv2.DCT_ROWS)

plt.subplot(2,2,1)
plt.imshow(dct1,cmap='gray',vmin=0, vmax=255)
```

```
plt.title('DCT_1 Original size')

plt.subplot(2,2,2)
plt.imshow(dct2,cmap='gray',vmin=0, vmax=255)
plt.title('DCT_2 Original size')

plt.subplot(2,2,3)
plt.imshow(dct1[0:100, 0:100],cmap='gray',vmin=0, vmax=255)
plt.title('DCT_1 100,100')

plt.subplot(2,2,4)
plt.imshow(dct2[0:100, 0:100],cmap='gray',vmin=0, vmax=255)
plt.title('DCT_2 100,100')




#
# End Script
```

Out[4]: `Text(0.5, 1.0, 'DCT_2 100,100')`



STEP IV: Apply DCT and inverse DCT

In this step,

- Find *inverse discrete cosine transform* using `idct = cv2.idct()` for `dct1` image
- Convert the results back to `np.uint8()` after applying `idct`
- Plot your results with Title of:

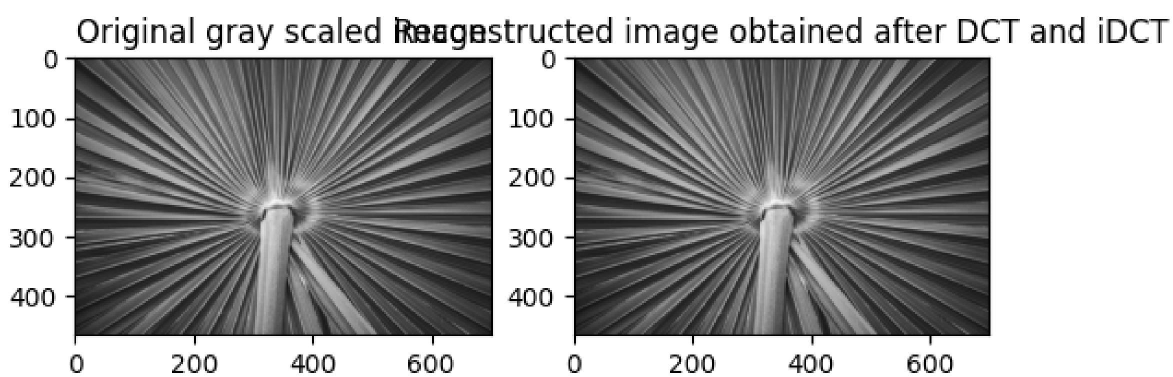
a. *Original* gray scaled image in `plt.subplot(1,2,1)`

b. *Reconstructed* image obtained after DCT and iDCT in `plt.subplot(1,2,2)`

using `plt.imshow()` using `cmap='gray'`, `vmin=0`, `vmax=255` as additional options

In [5]:
```
# Start Script below
#
# YOUR CODE HERE
idct = cv2.idct(dct1)
idct_reconstructed = np.uint8(idct)

plt.subplot(1,2,1)
plt.imshow(image_gray,cmap='gray',vmin=0, vmax=255)
plt.title('Original gray scaled image')

plt.subplot(1,2,2)
plt.imshow(idct_reconstructed, cmap='gray',vmin=0, vmax=255)
plt.title('Reconstructed image obtained after DCT and iDCT')


print(idct_reconstructed.dtype)

#
# End Script
```

```
uint8
```



STEP V: Compare the Result

In this step,

- Calculate the PSNR of both images using `psnr = cv2.PSNR(original_gray_scale_image, idct_image)`
- Print Image Type and Shape for both original and reconstructed images

In [6]:
```
# PSNR
psnr = cv2.PSNR(image_gray,idct_reconstructed)
print(f"The PSNR  =  ", psnr)
print("")
print(f"Original Gray Image Type                =  ", image_gray.dtype)
print(f"Aftre DCT and iDCT Gray Image Type      =  ", idct_reconstructed.dtype)
```

```
print(f"Original Gray Image Shape                =  ", image_gray.shape)
print(f"Aftre DCT and iDCT Gray Image Shape      =  ", idct_reconstructed.shape)
```

```
The PSNR  =   50.600550998808146

Original Gray Image Type                =   uint8
Aftre DCT and iDCT Gray Image Type      =   uint8
Original Gray Image Shape               =   (466, 700)
Aftre DCT and iDCT Gray Image Shape     =   (466, 700)
```

In [ ]:

```

```

End of Activity

DO NOT Run the notebook further